TYNE & WEAR



ATARI **8 BIT**

USER GROUP

Issue 22

July/August 1996

Reminder

TWAUG NEWSLETTER is published bi-monthly, around mid-month of (Jan, Mar, May, July, Sept and Nov.)

It is printed and published by TWAUG, no other publishing company is involved.

Opinion expressed by authors, in this newsletter, is their own opinion and do not represent the views of TWAUG.

The Atari Fuji symbol and Atari name are the trademarks of Atari Corporation. The Fuji symbol on the front cover , is for informational purpose only.

TWAUG is entirely independent and is in no way connected with Atari Corporation or any associate company.

Contacts:

Alan Turnbull on: 01670 - 822492

or Max on: 0191 - 586 6795

Postal address:

TWAUG

PO BOX 8,

WALLSEND

TYNE & WEAR

NE28 6DQ

NEWS & RUMOURS

Some late news from the USA.

by David Gostl (AAAUA)

This news arrived too late to include in last issue of TWAUG.

Atari Corp said it agreed to merge with JTS Corp. This is part of the text of the announcement.

SUNNYVALE, California

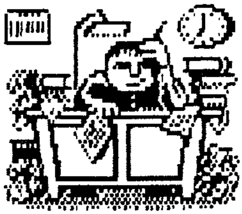
(BUSINESS WIRE)-- Feb. 13, 1996

Atari Corporation (AMEX: ATC) and JTS Corporation today agreed to merge the two companies. Atari is the pioneer in multimidia video entertainment and JTS is a manufacturer of computer disk drives.

"This merger puts us in a great position to capitulize on a very experienced management team and a rapidly growing disk drive market. JTS is using innovative technology, particularly in the 3 inch disk drive market and we are excited about its prospect," said Jack Tramiel, chairman of Atari.

"Our partnership gives us the ability to expand our capabilities and pursue new opportunities," said Tom Mitchel, president and chief executive officer of JTS.

TWAUG NEWSLETTER



PUBLISHING!

This new look newsletter is set up with the Desktop Publishing program "TIMEWORKS 2", on the Mega 1 ST with 4 meg memory. Files are converted and transferred to the ST with the Black Box transfer utility, as ASCII files. Those files are then imported into the DTP and printed with the Star/LC24-100 at 360dpi, with excellent result.

TWAUG

NEEDS  YOU

TWAUG subscriptions

Home .... 1 Copy £2.50
- DO - .... 6 Copies.. £12.50
Europe ..... 1 Copy £2.50
- DO - .... 6 Copies.. £13.50
Overseas.. 1 Copy £3.50
-- DO --..... 6 Copies.. £16.00

Issue 23 is due mid-September

ISSUE CONTENT

DON'T LET BASIC BUG YOU

Programming made easy - Part V of MIKE BIBBY's guide through the micro jungle

STRAIGHT to work in this issue. Have a look at Program I. I don't think it should cause you too many problems.

We're just assigning the values 1, 2, 3 to three numeric variables, NUMBERONE, NUMBERTWO, NUMBERTHREE, and printing out the value of the variable immediately after each assignment.

```
10 REM PROGRAM I
20 PRINT CHR$(125)
30 NUMBERONE=1
40 PRINT NUMBERONE
50 NUMBERTWO=2
60 PRINT NUMBERTWO
70 NUMBERTHREE=3
80 PRINT NUMBERTHREE
```

Program I

The end result is that:

1

2

3

appears on our screen. A long-winded way of doing things, I admit, but easy enough to follow.

Program II is a different kettle of fish, but, believe it or not, its output is exactly the same as in Program I.

It's sensible enough down to line 40. We clear the screen in line 20, assign the value 1 to numeric variable NUMBER in line 30, then PRINT NUMBER in line 40.

When equals doesn't make

Line 50 looks decidedly odd, though:

```
50 NUMBER=NUMBER+1
```

How can a number be equal to itself plus one? That's what line 50 seems to be saying, after all.

all things equal ...

The fact is that equals sign doesn't mean equals here - it just tells the computer to do something. The equals sign instructs the computer to do whatever task is given on its right and then label the result of that task with the label on its left.

DON'T LET BASIC BUG YOU continued

```
10 REM PROGRAM II
20 PRINT CHR$(125)
30 NUMBER=1
40 PRINT NUMBER
50 NUMBER=NUMBER+1
60 PRINT NUMBER
70 NUMBER=NUMBER+1
80 PRINT NUMBER
```

Program II

In this case the micro interprets line 50 as starting on the right of the equals sign, take the value labelled by NUMBER and add one to it. Then label the answer with the variable NUMBER. The micro doesn't bother that the same label has been used on both sides, it just updates NUMBER with its new value.

The practical effect of line 50 is to increase NUMBER by one -to two. Line 60 then duly prints out this new value of NUMBER.

Line 70 is identical to line 50. Starting at the right of the equals sign it takes the value of NUMBER, increases it by one, then re-labels it with NUMBER. That is, NUMBER increases from two to three. Line 80 then prints out the new value of NUMBER.

The thing to remember is that the equals sign doesn't mean equals - it means assign. You "do" what's on the right of the equals sign, and then assign the result to the label on the left.

Let's take this idea a little further. Have a look at Program III. The first five lines should be fairly familiar.

When we run it the screen clears, line 20, we set NUMBER equal to zero, line 30, increase it by one, line 40, and then print it, line 50. Since NUMBER was zero, and we've increased it by one, the result will be that 1 appears on the screen.

Once the program's done this, we come to line 60 which reads:

```
60 GOTO 40
```

As you'll recall, the GOTO 40 tells the micro to make line 40 the next line it does. This increases NUMBER by one, as we've seen, so NUMBER takes the value two. Line 50 then prints out the 2 and we encounter line 60 again.

This sends us back to line 40, which increases NUMBER. Line 50 prints out the new figure, 3, then we're back at line 60, which takes us to line 40, which increases NUMBER, and so on.

DON'T LET BASIC BUG YOU continued

I think you can see that the program will produce the steadily increasing

```
10 REM PROGRAM III
20 PRINT CHR$(125)
30 NUMBER=0
40 NUMBER=NUMBER+1
50 PRINT NUMBER
60 GOTO 40
```

Program III

sequence of numbers 1, 2, 3, 4, 5, 6, 7 and so on.

Try running it and see. You'll probably be glad to know that the way to break out of the program is by aptly named Break key.

If you simply want to freeze things for a moment while you examine the output, press the Control and the 1 (one) key at the same time - we write this as "press Control+1". To restart things simply press Control+1 again.

As we've mentioned, when a program keeps going round in circles like this, we call it a loop. We can then make statements such as NUMBER increases by one each time round the loop.

If you press Break, or freeze it quickly enough after the start of the program, you'll see that the first value of NUMBER printed out is one, and not zero as you might think. All right, we assigned zero to NUMBER in line 30, but we increased it by one immediately, before ever printing it out.

But what if we wanted the zero printing out? Well, a sneaky method would be to make line 30 of Program III:

```
30 NUMBER=-1
```

What happens here is that line 40 immediately increases NUMBER by one, to make it zero (-1+1=0). Line 50 then prints it out.

```
10 REM PROGRAM IV
20 PRINT CHR$(125)
30 NUMBER=0
40 PRINT NUMBER
50 NUMBER=NUMBER+1
60 GOTO 40
```

Program IV

Another way round is just to swap lines 40 and 50, so we PRINT before we increase NUMBER. This is what I've done in Program IV. Try running it and you'll see - if you're quick enough - that 0 does appear on your screen.

Actually we don't need to go up in steps of one. Have a look at line 50

DON'T LET BASIC BUG YOU continued

of Program V:

```
10 REM PROGRAM V
20 PRINT CHR$(125)
30 NUMBER=0
40 PRINT NUMBER
50 NUMBER=NUMBER+3
60 GOTO 40
```

Program V

NUMBER=NUMBER+3

Remembering that micros start on the right of the equals sign, the Atari takes the value labelled by NUMBER, adds three to it, and gives the result the label NUMBER. The effect is that line 50 increases the value NUMBER by three each time round the loop, so numbers are printed out in steps of three.

Nor do the numbers always have to be getting larger. Program VI, as

```
10 REM PROGRAM VI
20 PRINT CHR$(125)
30 NUMBER=1000
40 PRINT NUMBER
50 NUMBER=NUMBER-1
60 GOTO 40
```

Program VI

you'll see without too much difficulty, starts at 1000, then prints out 999, 998, 997 and so on. The crux here is line 50:

50 NUMBER=NUMBER-1

Try running it if you won't take my word for it!

In fact we can write a general program that will start at any number and go up or down in whatever steps we want by using the INPUT statement we met in last issue. Program VII does the trick.

First of all the program asks us the number we want to start printing from - which we label START. Notice how line 30 politely prints out a message to tell us what we're supposed to INPUT. Line 40 does

```
10 REM PROGRAM VII
20 PRINT CHR$(125)
30 PRINT "Number to start at";
40 INPUT START
50 PRINT "Increment of";
60 INPUT INCREMENT
70 NUMBER=START
80 PRINT NUMBER
90 NUMBER=NUMBER+INCREMENT
100 GOTO 80
```

Program VII

DON'T LET BASIC BUG YOU continued

the actual INPUT, labelling it as START.

Lines 50 and 60 then prompt for, and INPUT the increment, or step, by which we want the numbers to go up, labelling it INCREMENT.

We then get down to business. Line 70 assigns the value of START to NUMBER. We then print this value of NUMBER in line 80, so we're off to a good start, if you'll pardon the pun.

We then have to increase NUMBER by the value of INCREMENT to get the next value. Line 100 then jumps back to 80, which then prints the updated value. Line 90 then increases it again by INCREMENT and so on.

If you have difficulty visualising this try substituting sets of real numbers for START and INCREMENT and see what happens as you go round the loop.

For instance, if you mentally input 25 for START and 5 for INCREMENT, line 70 would give NUMBER the value 25, which line 80 would then print out. Line 90 would then add 5 to this, giving NUMBER the value 30 then we'd loop back to 80 via 100. The figure 30 would then be printed out, line 90 would increase it by 5 again, and so on.

All well and good, but having to escape from these loops by pressing Break isn't very elegant is it? Ideally the program should stop of its own accord. In other words, we should give it a condition to finish on. the loops we've met so far haven't had any finishing condition so they're known as unconditional loops.

We need to create a conditional loop, and Program VIII shows us how we go about it.

```
10 PROGRAM VIII
20 PRINT CHR$(125)
30 ? "I'll keep on going until you
enter 999"
40 INPUT NUMBER
50 IF NUMBER=999 THEN STOP
60 GOTO 30
```

Program VIII

The idea is that we keep on looping round, printing out the same inane message, until we enter 999. That is, the condition for ending the loop is that we INPUT 999 - any other number will cause the loop to be repeated.

Let's see how we achieve this: Line 20 clears the screen, then line 30

DON'T LET BASIC BUG YOU continued

prompts for the INPUT, saying the program will keep on going until 999 is entered.

(In case you're wondering what the ? is in line 30, it's the Atari's abbreviation for PRINT. If you want to substitute ? for PRINT, then go ahead. I prefer the clarity spelling it out gives you.)

Line 40 then INPUTs into the variable NUMBER. Line 50 is the heart of the matter - this is where we test for our condition. It reads:

50 IF NUMBER=999 THEN STOP

This uses the IF ... THEN statement, one we haven't met before. It reads:

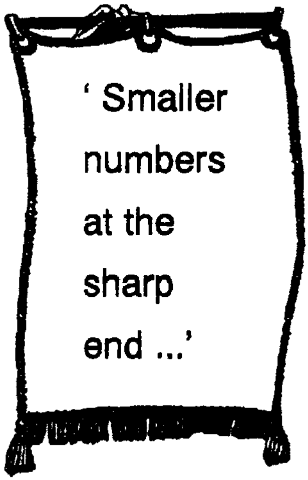
☆ IF some condition is TRUE, do what follows the THEN. IF that condition isn't TRUE, ignore what's after the THEN and carry on with the next line.

In this case of line 50, this boils down to:

☆ IF NUMBER does indeed equal 999 THEN STOP. IF NUMBER isn't 999, THEN drop through to the next line - line 60 in this case.

So if when prompted by line 30, we'd INPUT a value of 999 for NUMBER, we would do what's after THEN and STOP.

We haven't met STOP before, but I don't think you'll be too surprised to learn that it stops the micro dead in its tracks. It also prints out a message indicating the line it was stopped at. In this case it would be STOPPED AT LINE 50.



If, however, we entered a value for NUMBER other than 999, our condition hasn't been met, so we carry on with the next line of the program, line 60, which then sends us back to our prompt for INPUT again.

In other words, the IF ... THEN statement ensures that we keep on looping until we enter the number 999. We've got ourselves a conditional loop!

Actually there are other ways of stopping it, such as pressing Break or entering a word when it's expecting a number. For the moment we'll assume that you're

DON'T LET BASIC BUG YOU continued

good-mannered enough to avoid this. Later on we'll see how to "idiot proof" our input, as it's known.

Try running Program VIII, and enter 999 to stop it. Then enter:

CONT

The program will restart - CONT stands for continue. The same trick works after you've pressed Break.

The IF ... THEN statement isn't too hard to use, just remember:

- The condition you're testing for comes directly after the IF.
- You can put any valid Basic instruction after the THEN.
- The instruction after the THEN is only carried out if the condition has been met - that is, if it's true.
- If the condition is not true, the micro ignores what's after the THEN and continues with the next line of the program.

Take a look at Program IX now. This does exactly the same job as Program VIII but in a different way. This time we test to see if the number is NOT equal to 999, and, if this is so, we loop back to our prompt for INPUT. Lines 20 to 40 are identical. The vital bits are lines 50 and 60.

Line 50 reads:

```
50 IF NUMBER<>999 THEN GOTO 30
```

Here our condition is that NUMBER

```
10 REM PROGRAM IX
20 PRINT CHR$(125)
30? "I'll keep on going until you enter
999"
40 INPUT NUMBER
50 IF NUMBER<>999 THEN GOTO 30
60 END
```

Program IX

isn't equal to 999. That's what <> means - not equal to. And if our condition's true - that is, NUMBER isn't equal to 999 - we THEN loop back to line 30.

On the other hand, if the condition in line 30 isn't met - that is NUMBER is 999 - we simply drop through to line 60, which reads:

60 END

We met this before. As its name suggests, it simply ends proceedings, this time without any message, unlike STOP.

The not equal to symbol, <>, may be new to you. It's just one of a set of inequalities, as they're known, that come in very useful in combination with IF ... THEN statements. Table I summarises them.

DON'T LET BASIC BUG YOU continued

If you're anything like me, you'll get confused between > and <. The trick is to remember that, for both symbols, the larger number goes opposite the bigger end of the symbol, whereas the smaller number goes opposite the sharp, or smaller end. It may not be the way Einstein remembered it, but it's good enough for me!

Program X uses what we've learned about IF ... THEN statements, as they're known, to add a finishing point to Program VII. Lines 10 to 40 are identical. We clear the screen and INPUT a value for START, the number we start from.

Lines 50 and 60 then prompt for and INPUT the number we wish to end at, FINISH. 70 and 80 then INPUT INCREMENT, the value of our step. As in program VII, the value of START is assigned to NUMBER, line 90, and then printed, line 100. Next, we increase the value of NUMBER by INCREMENT and store it in NUMBER again at line 110.

Line 120 is the crux:

```
120 IF NUMBER>FINISH THEN END
```

What this says is, if we've incremented NUMBER past FINISH end the program here and now. If not, then continue with the next line, which will loop back to 100 and print the newly-incremented number.

Notice that if we have exceeded the limit we don't actually print that value of NUMBER since we don't loop back to the PRINT statement of line 100.

Experiment with different values and see if you can undestand what's happening. This program is quite fundamental, and we'll be using its ideas a lot, so it's worth an effort.

```
10 REM PROGRAM X
20 PRINT CHR$(125)
30 PRINT "Number to start at";
40 INPUT START
50 PRINT "Number to finish at";
60 INPUT FINISH
70 PRINT "Increment of";
80 INPUT INCREMENT
90 NUMBER=START
100 PRINT NUMBER
110 NUMBER=NUMBER+INCREMENT
120 IF NUMBER>FINISH THEN END
130 GOTO 100
```

Progam X

For instance, if you start at 25, set the finish to 35 and prescribe a step of 5, you'll get:

25

30

DON'T LET BASIC BUG YOU continued

35

on your screen.

However a start of 50 and a finish of 55 with an increment of 3 will result in:

50

53

You won't see a 56, because it exceeds our limit.

What happens if the start and finish numbers are the same, or the increment is negative? In fact, will the program work with negative numbers? And what would happen if we choose an increment of zero? Find out!

- Well that's all for this issue. In next issue we'll be continuing with loops, but in an entirely different way.

Inequality	Meaning	Example
=	equals	7=7 is true, 8=7 is false
>	greater than	7>5 is true, 3>4 is false
<	less than	3<4 is true, 7<7 is false
< >	not equal to	4< >5 is true, 4< >4 is false
>=	greater than or equal to	6>=5 is true, 6>=9 is false
<=	less than or equal to	7<=7 is true, 7<=6 is false

Table I: Inequalities

The 130XE - 576K upgrade

by Scott Peterson.

Copyright © 1986, released to the public.

Here we go again, this time I recommend you have some electronics experience if you wish to preform the upgrade. Some of the work is duplicated from the 320K upgrade so 320XE owners will not have as much work to do. One other point, when in the 576K mode you MUST use some sort of basic cart. as you lose the internal basic, this is only in the 576K mode, in the 130XE mode internal basic will function normally.

TOOLS NEEDED;

To preform this upgrade you need the following;

- ☆ Low wattage fine tip soldering iron. Vacuum de-soldering tool (like Radio Shack PN#64-2098).

THE 130XE - 576K UPGRADE continued

- ☆ Some 30-gauge wire(Radio Shack PN#278-501).
- ☆ #2 phillips head screwdriver.
- ☆ Heat-shrink tubing, 1/8 in. Dia.
- ☆ Also a pair of small needle-nose pliers and a small flat tip screwdriver are handy.

PARTS NEEDED;

- ☆ Z1 74LS158
- ☆ Z2-Z17 41256(150ns.)
- ☆ Z18 74LS138
- ☆ Z19 7432
- ☆ R1-R2 33 ohm 1/4 watt resistor.
- ☆ S1 Micro-mini DPDT switch(like Radio Shack PN#275-626)

Remove the 130XE case and metal RF shield to get down to the mother board. (320XE users go to step two).

STEP ONE:

Now de-solder and remove the eight ram chips U26 thru U33 (MT4264). They are the row closest to the TV RF module (do NOT use solder wick, the circuit board of the 130XE has very weak runs and they will pull loose if not completely de-soldered). Replace these with the 16 pin low profile sockets.

Take a piece of wire approx 12 in. long and run a jumper from pin 1 of each socket to the next. When you are done the wire should be attached to pin 1 of each of the new sockets and you should have about 6 inchs left over. Do this on the rear of the mother board and then snake the wire thru the large hole near the ram chips.

Next, desolder and remove U23 (CO14795), and replace it with a 40 pin socket. Bend up pins 15 and 16 and insert it in the socket you just installed.

Take Z1 (74LS158) and break off pins 5,6,7,9,10,11,12,13,14. Bend up the other pins on it except 8 and 16. Put this "piggy back" on top of U20 (HD14050, or 4050 -located just to the right of C50) and solder pins 8 and 16 of Z1 to pins 8 and 16 on U20. Now take a short jumper from pin 15 on Z1 to pin 8 of Z1.

Take a piece of wire about 4 in. long, solder one end to pin 30 on the chip marked "CO14805" on the mother board, and the other end to pin 1 on Z1.

Next solder a wire to pin 15 (one of the two you bent out) of U23 and connect the other end to pin 2 on Z1. Solder a wire to pin 16 on U23 and connect the other end to pin 3 on Z1.

THE 130XE - 576K UPGRADE continued

Take R1 (33 ohm) and trim the leads to about 1/4 in. Take the wire you connected to pin 1 of the ram chip sockets and solder it to one end of R1, solder the other end of R1 to pin 4 on Z1.

STEP TWO:

Slide the mother board back into the bottem half of the plastic case (do not use the RF shield, you must be able to get at the mother board), and attach the keyboard. It will rest above the mother board without touching it. Test all 41256 ram chips by putting one set of 8 in the sockets and using the handlers (or DOS's), and then the next.

After testing all ram chips remove them all from the sockets, and take 8 of them and cut about half of pin 15 off of each one. Only the "fat" part of pin 15 should be left. After doing this you have to "piggy back" the 8 256K ram chips with the short pin 15's on top of the other 8 256K ram chips. Now solder all the pins together on the stacked ram chips except for pin 15, it should not be touching the other pin 15, make sure you have them going pin 1 to 1, pin 2 to 2, ect. When you get done you will have 8 sets of Piggy backed 256K ram chips.

Now take a piece of wire about 16 in. long and run a jumper from pin 15 to the next one on all the top 256k DRAM's, leaving about 1 inch between each ram chip. Put these stacked ram chips into the 8 sockets you installed earlier.

Take Z18 (74LS138) and bend up all the pins except 8 and 16, cut the pins you bent up in half so only the fat part is left, and solder pins 8 and 16 to pins 8 and 16 of the other 74LS138 right below the U23 (CO14795).

Take Z19 and bend up all pins except 7 and 14, once again cut all the pins you bent up in half and solder pins 7 and 14 to pins 7 and 14 of the 74LS08 right below U23.

Take the wire you jumpered earlier to pin 15 of Z10 thru Z17 (the upper row of 256K ram chips) and go out 2 in. and cut the wire, now install R2 (33 ohm) between this cut. Place a piece of heat shrink tubing over R1 and make sure no wire is exposed and heat it with a lighter. Take the other end of this wire and connect it to Z18 pin 14.

Find the 2 33 ohm resistors just to the right of U28 (one of the ram chips you socketized). The upper one of the 2 is R111, desolder the right leg of it and bend it up. Take a piece of wire and solder it to the

THE 130XE - 576K UPGRADE continued

land where you just removed the leg of R111. Connect the other end to Z18 pin 4. Trim back the leg of R111 and solder a wire to it, slip a piece of heat shrink tube over it and heat it up.

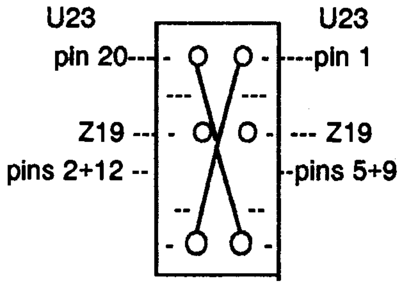
Now connect the other end to Z18 pin 12. Take a short wire and run a jumper from pins 1 and 16 of Z18. Take another short wire and connect a jumper from pins 3, 5, and 8 of Z18. Now connect a wire from Z18 pin 2 to Z19 pin 3.

Find the wire you installed from U23 pin 15 to Z1 (74LS158) pin 2 and desolder it from U23. Take it and reconnect it to Z19 pin 11.

Ok, now pry U23 (CO14795) back out of the socket and bend up pin 11, plug it back in. Run a jumper from pins 1 and 4 of Z19, and another jumper from pins 10 and 13 of Z19. Connect a wire from U23 pin 11 to Z19 pin 1, and from U23 pin 15 to Z19 pin 13. Now connect a wire from Z19 pin 8 to the right side of the 3.3K ohm resistor marked R206 (located at the bottom right of U23). Connect a wire to Z19 pin 6 and run it to pin 18 of U3 (CO61618).

Now comes the tricky part, drill a small hole (1/4 in. or so, depending on the switch size) at the rear right of your 130XE. Take the small DPDT switch (S1) and install it in the hole. Now connect it but (make sure the switch DOESNT have a center off position);

S1 (rear)



Note: where the wires cross in the middle, they are NOT connected. Make the connection from the switch to U23 on the rear of the mother-board.

Well that's it (thank god).

Now re-assemble the computer, being carefull not to break any wiring going to the switch. You should now have in one switch position a 100% compatable 130XE, and in the other you have a 576K 130XE that does not have Antic memory enhance mode and also can NOT use internal basic. In the 130XE mode you gain 64K as bit 6 of the PIA can still be used. The following page list of the bit table and numbers to be used in location

THE 130XE - 576K UPGRADE continued

54017 (PORTB).

Once again, if you need help call the Peanut Gallery BBS (408)-384-3906. If you want a mailer of all the upgrades I have as well as a disk with handlers, source codes, ect. send a money order (please, no checks) for $10.00 to:

Scott Peterson

P.O.Box 33

Ft.Ord CA. 93941-0033

This includes the 800 288K upgrade by D.G.Byrd, the 800XL/256K (C.Burchholz), the 130XE/320k upgrade and anything else I finish. Good luck, and have fun.

Memory Control Register 54017(D301) 130XE in 576K mode.

Bit 7 6 5 4 3 2 1 0

D a b C c d e R

D=0 enable diag. ROM

R=1 enable OS ROM

C=0 enable extended memory

abcde=

memory control bits.

Bank #	Control # (dec)	Hex
Bank 0	129	81
Bank 1	131	83
Bank 2	133	85
Bank 3	135	87
Bank 4	137	89
Bank 5	139	8B
Bank 6	141	8D
Bank 7	143	8F
Bank 8	161	A1
Bank 9	163	A3
Bank 10	165	A5
Bank 11	167	A7
Bank 12	169	A9
Bank 13	171	AB
Bank 14	173	AD
Bank 15	175	AF
Bank 16	193	C1
Bank 17	195	C3
Bank 18	197	C5
Bank 19	199	C7
Bank 20	201	C9
Bank 21	203	CB
Bank 22	205	CD
Bank 23	207	CF
Bank 24	225	E1
Bank 25	227	E3
Bank 26	229	E5
Bank 27	231	E7

THE 130XE - 576K UPGRADE continued

Bank 28	233	E9
Bank 29	235	EB
Bank 30	237	ED
Bank 31	239	EF

There is a version of MYDOS to support this mod, its called 4.1A and will run up to 32 16K banks. At this time ICD is working on a RD.COM file to support this.

Also I have written a machine load tester that will load and test all 32 banks of memory to insure that they are there and work. Wonder how long it will take Jay Torres to copy this one.

Good luck Scott Peterson

This is the last upgrade article that has been supplied to TWAUG.

Atari Portfolio Club

Are you a Portfolio user? If so why not join the new Portfolio Club, membership is free. Contact Paul Finch, 16 Cedars RD, Morden, Surrey, SM4 5AB
Enclosing SAE for more details and entry form.

- B - TAPE -

Configuration file

The next file (after the DOS) must be a configuration file. This file must be named "CONFIG.B", and it is a text-file that contains the commands for initializing the system after boot up. (Like the "STARTUP.BAT" file on disk.) This file should be shorter than 500 bytes - otherwise an "Out of memory" error may occur with the internal Atari-Basic active. You can create the configuration file simply by copying it from the "E:" device - use the command "COPY E: B:CONFIG.B", and then type the contents of the file (use <CONTROL>+<3> to finish). The configuration file is interpreted by the Boot loader. This is necessary to allow reading of high-speed tape files before installing the "BTAPE" command, but there are also some limits: The configuration file may contain only resident commands, and the commands "COPY", "DATE", and "TIME". No commands in the configuration file may be started from other devices than the high-speed tape system (but parameters may work with other devices too). The used commands may not use memory above the

------ B - TAPE ------

address $9000 (MEMHI), because the Boot loader itself uses this memory. It may occur, that a tape-file must be used as a parameter of some command before installing the "BTAPE" driver. In this case, you should call the tape-files as "Q:filename" - the request will then be executed by internal routines of the Boot loader. The device handler "Q:" exists only while the configuration file is executed, and it can only read tape files. When the configuration file is finished, the "Q:" handler disappears together with the Boot loader. The wildcards are supported by "Q:", but you can't use it with the "COPY" command (in this case, COPY will try to get the actual filename using the "DIR" function - not supported by "Q:"). When the last line of configuration file begins with a "+" character, the rest of the line will be executed in the CP. In this way, you can automatically start any program after boot, or run a batch file (previously copied to a Ramdisk) to continue with the initialization of the system.

After the configuration file, you should record all the files necessary for executing the configuration. (That's for each program-file line and any files used as parameters.)

Example: To simply boot a DOS and BTAPE, the configuration file "CONFIG.B" will look like this:

BTAPE /O

In this case, you should record the following files to the boot-tape:

The Boot loader (see "TAPE-BOOT") XBW130.DOS (or other compatible DOS) CONFIG.B (see above) BTAPE.COM

Another example: Boot a DOS, install RAMDISK (on the Atari 130XE computer) and then BTAPE. Place the "BTAPE.COM" file into the Ramdisk to allow free installing/removing of BTAPE depending on memory usage of application programs. The configuration file will be following:

```
RAMDISK 1F Q:B_130XE.RD
COPY Q:BTAPE.COM D1:
+D1:BTAPE /B
```

You should record the following files:

The Boot loader (see "TAPE-BOOT") XBW130.DOS (or other compatible DOS) CONFIG.B (see

------B - T A P E ------

above) RAMDISK.COM B_130XE.RD COPY.COM BTAPE.COM

LOADING GAMES

Under B-TAPE, you can load almost any games (and other programs like demos etc.) as long as they are stored in a binary file. Every programs started under a DOS or Micro-DOS are compatible, and also many tape-versions uses this format (the games with a short loader, mostly displaying a "!" in the bottom-right corner of the screen - you should copy only the main file without of the loader).

The MICRO-B command

Some games may be started under BW-DOS, but mostly the DOS is too long to fit into memory together with the game. With a disk drive, we are using different "Micro-DOS" programs in this case - the same tool for tape is called "MICRO-B"

MICROB (C|T|S) (External command) This command records the MICRO-B loader to tape. With the parameter "C" it creates a common boot-file (in the "C:" format), that may be loaded simply by holding down the <START> key during power-on. The "T" parameter gives the same loader in the format "TURBO 2000" - a format that is popular here in the Czech Republic. With the parameter "S", this command will directly start the MICRO-B loader.

Using the MICRO-B loader

The MICRO-B loader can only start a game or another program of the same type. No more support (like access to tape-files) is provided. The loader is easy to use. After starting it, any file in the B-TAPE format is accepted from tape. When any block is found, the program displays the filename, and asks you if you wish to load that file. When you answer "N", another block will be searched, otherwise the file will load.

When started, the MICRO-B loader executes a few actions to provide the best compatibility with game-programs. (Most of these actions are the same as under "Micro-SpartaDOS".) First, any DOS will be removed, and the Atari-Basic will be switched off (no need to hold down <OPTION> while booting). Then, MICRO-B clears the

------B - T A P E ------

memory, and sets a special mode for the <RESET> key, so most of the games will not be "reset-proof" under MICRO-B (the number of necessary "OFF-ON" switchings is greatly decreased).

The MICRO-B loader should load most of the file-based games. The only problem may occur with MEMLO - with MICRO-B version 1.0 the MEMLO value is $CAD.

The BINCOPY command

While loading games, demos etc., a problem with length of the gaps between blocks may occur. Many programs of the mentioned type contains different intros or depacking routines (for example the ones generated by "Super Packer"), that are executed between loading. This mostly causes the recorder to overrun the beginning of the next block, and so the user must rewind the tape a bit. Since this is really boring, the programs with such intros or depacking routines should be copied to tape with the "BINCOPY" command.

BINCOPY file (External command).

This program is very similar to "COPY". It only copies one file, and the destination is always "B:" - that's why only one parameter (source) is required. The limit of file-length while copying directly from tape to tape is the same as with "COPY". The command "BINCOPY" may copy any file, but the reason why it was written is for copying binary program files, especially games. In this case, the "BINCOPY" program checks the file-structure, and generates a long gap after every blocks, where a kind of intro etc. may occur (where an INIT vector is found). The games copied in this way are loading with no problems, never mind that intros etc. are present, and the loading-time is not increased like with the mode of long gaps.

TECHNICAL INFO

This chapter gives the necessary information for using the functions of BTAPE in application programs. It was written for programmers - if you don't understand it you can skip this chapter.

Functions of the "B:" handler

The main part of the B-TAPE system is handler "B:". This handler can open only one file at a time. No

------B - T A P E------

physical errors are returned, every actions like searching for the correct file, reading retries etc. are supported by the handler itself. You may only encounter errors 128, 138, 152, 161, 163, 166, or 168.

The "OPEN" function needs a filename in the same format, as with a disk drive (but without of path of course - the syntax is "B:name.ext"). The possible values of "aux1" and "aux2" (the second and third parameter of "OPEN" in Basic) are:

Aux1=4 - Reading a file. In this mode, it is possible to change the position in the file using NOTE and POINT functions. Aux1=6 - Reading of the directory. See "DIR" and "DIRS" commands for more info. Aux1=8 - Recording a file. Aux2=128 - Short gaps between blocks. Aux2=64 - Double blocks.

Both these functions of "Aux2" may be added together of course. These functions may also be selected by a parameter while installing the "BTAPE" command - in this case the "Aux2" parameter have no effect (it is ORed with the settings done during installation).

A few more functions are available in the same syntax, as with a disk drive: NOTE (XIO 38), POINT (XIO 37), and binary load (XIO 40). The functions NOTE and POINT are only possible while reading a file.

The handler "B:" allows also access to incomplete files (for example when recording overruns a part of another file on tape). It's possible to read any block, that is readable for the "DIR" function. The access to incomplete files is possible by executing OPEN and then immediately POINT. In this case, BTAPE searches directly the POINTed position. Vieweing incomplete files is easiest with the command "DUMP". You'll need to search the first readable block with the "DIR" function, multiply the serial number of that block by 1008 (the length of a block), and use the result as a file-position for DUMP - type "DUMP B:filename result". In the case that the last block of the file is missing, you need to abort the reading with the <BREAK> key.

SIO-level access

Using the vector "LSIO" in the "COMTAB" table (see the DOS manual), you can call directly the read/write routines of BTAPE. It will

------B - T A P E ------

not affect any function of the "B:" handler - you can even do it with a file open on "B:". On this level, the things like reading retries, sound signals etc. are not supported; only the reading routine is able to wait for the beginning of a block. The routines simply return errors like 140 and 143 when a physical error occurs. The starts/stops of the tape recorder are fully sopported by BTAPE also on the SIO-level - just call the LSIO vector, and BTAPE will do it for you. On the SIO-level, the whole physical blocks are transferred - including the internal header-bytes (see later). Recording the sound signal "start of file" is not available on the SIO-level. The use of BTAPE SIO-routines may be useful for example while creating a "comfortable" copier-software (handling two files with changing two tapes, verifying the tape etc.). Note that the used buffer may not be in the area of $4000-$7FFF, or above $C000. That's because the SIO-routines are time-critical, and so it's impossible to switch memory-banks for each byte (when BTAPE is installed in a Ramdisk bank, or under OS-ROM). It's also possible to install new SIO-routines into the LSIO vector - such routines will then be used by the "B:" handler. But you must know, that in this case the buffer used by the "B:" handler may also be in a bank, or under OS-ROM.

The settings before calling LSIO are as follows:

DDEVIC ($300) = $61 (B-TAPE) DUNIT ($301) = 1 (unused) DCOMND ($302) = "R" (read) or "W" (write) DSTAT ($303) = 64 (read) or 128 (write) DBUFL/H ($304) = Address of buffer DTIMLO ($306) = $FF (unused) DBYTL/H ($308) = Length of buffer DAUX1 ($30A) = 0 (unused) DAUX2 ($30B) = The value "aux2"

The length of the standard B-TAPE blocks is $401. You can transfer blocks of any length, but then you can't access it with the "B:" handler of course. The value "aux2" have the same function as with OPEN in Basic (see above), but on the SIO-level the value is not affected by the parameter used while installing BTAPE. The speed is defined while installing BTAPE, and may not be changed (excepting that you call the "BTAPE" command again). The handler "B:" is recording the first block of each file

------B - T A P E ------

twice, with aux2=0 (to provide a long gap between the blocks). The next blocks are doubled (when necessary) using the corresponding aux2 value.

Physical format

The following part of this text describes the physical format of data on tape. It's only in case someone is interested - the described format is supported by SIO-routines of BTAPE, that are also accessible for other programs (see above). The signal recorded on tape describes single bits with different lengths of impulses. This system came from TT-DOS, and it was used because of compatibility with older Czech Software. The origin of this system is in the line ZX-Spectrum - TURBO 2000 - TT-DOS - B-TAPE. The used format is not supported by the Atari hardware, so all decoding is software-based. That's why the screen must be off during the I/O operations. Recording to tape uses POKEY-timers, so the timing of created signal is perfect (that's different from older "Turbo" systems like TT-DOS). Reading of this format is possible only with a tape recorder modified for "Turbo". This upgrade is activated by storing $34 at $D302 and $D303. Then, the input data-line reads directly the signal from tape. Each bit is recorded as one impulse followed by a pause of the same length. While reading, the length of impulse and pause together is significant. The exact length of these impulses depends on the used transmission speed. The impulses for a bit with value "1" are exactly two times longer, than the impulses for a "0" bit. A block begins with a lot of "1" bits. This signal is used for synchronization of the reading routine, and also to recognize the used transmission-speed. The length of this signal (0.5-1.5 sec - depending on the used mode) also helps with the compensation of different times used for interpretation of data between blocks. A single "0" bit is following. This bit indicates the begin of actual data. Then the data-bytes are stored, highest bit first. At the end, one internal check-byte is stored - this byte never appears in the data buffer. The value of this byte is a combination of all data-bytes using the function Exclusive-OR (EOR in

------B - T A P E ------

assembler). The signal of a block is finished with a single bit more - the value of this bit isn't defined. This bit is necessary for a safe interpretation of the last recorded byte - this bit closes the pause after last significant impulse. The polarity is not defined - the bits may be stored as "high-low" or "low-high" (so copying the tape using different audio-devices makes no problem at this point). When the recording of a block begins immediately after turning the recorder on, a short pause is added before the start of the signal. This avoids affecting the synchronization-signal by "waking up" the recorder, or by an old signal (because of the delay between the clearing and recording heads). Affected signal leads to errors in recognized transmission-speed, and so decreases safety of the transfer. The melody-signal "start of file" is just a sound-effect for better orientation, it has no sense for the reading routine.

Logical format

Each block under B-TAPE is 1025 ($401) bytes long, and it contains 1008 ($3F0) bytes of data, and 17 ($11) header bytes. The info about the structure of a block is necessary while working on SIO-level. The bytes in a block are:

0 - Serial number of the block in file. This number may be between 1 and 255, the block 0 is not used.

1 - Recording mode. This is a copy of "aux2" value used while recording the file - including the ORed settings defined while installing BTAPE. This byte is used by the "DIR" function to recognize and show files with double-blocks.

2,3- Position of the last significant byte in the block (only the lower 11 bits). Every block accepting the last one has $400 here (Incomplete blocks in the middle of a file are not allowed - because of the NOTE and POINT functions). When the length of file is not equal to zero, empty blocks (value 16) are never present. The last block of a file has the highest bit (bit 7 in the byte 3) set to "1". The unused bits should always be zero.

4 - Unused byte (reserved). Under B-TAPE it's always zero, and it should also be with any other programs (working at SIO-level for example). When using the old TT-DOS, you may encounter almost any value here. The value is always

------B - T A P E ------

the same for every block in a file.

5 - The random number. This number is created (randomly) during the OPEN operation, and is always the same for every block in the file. It is used to recognize and reject blocks from other files even if the filenames are identical. The old TT-DOS doesn't create the random numbers.

6-16- The filename (8 characters for the name and 3 characters for the extension).

17-1024- Data bytes. The unused bytes in the last block of a file are always zero under B-TAPE, but while working with the old TT-DOS, these bytes are not defined.

COMPATIBILITY

In this chapter you'll find some info about compatibility between B-TAPE and selected programs. The most common problem with compatibility will probably be a collision in memory - see also the chapter "Installation" for more info.

SpartaDOS

B-TAPE is compatible with SpartaDOS versions 3.2d, 3.2e and 3.2f (other versions weren't tested). While installing BTAPE under OS-ROM with SpartaDOS, you must use the parameter "/OX" (the function AINIT will not be available and the errors in CP will be shown as numbers only). The internal command "COPY" in SpartaDOS is not fully compatible with B-TAPE. It can't carry original filenames while copying from tape (you must type both source and destination filenames), and it cannot copy directly from tape to tape. When necessary, you can use the external "COPY" from BW-DOS by typing "COPY.COM" instead of "COPY". But please don't use the "COPY" from BW-DOS for disk-to-disk, copying under Sparta - this leads to incorrect date/time information on the copied files.

Turbo Basic

To get the popular Turbo Basic working with B-TAPE, you must install the "BTAPE" command into a Ramdisk bank (use the parameter "/B"). This works only with min. 128kB RAM of course.* On a 64kB system, the Turbo Basic may not work with B-TAPE installed at the same time. The only solution in this case is the called "Turbo Basic

-B - T A P E

3.2q" (a version for SpartaDOS), with SpartaDOS 3.2 and "BTAPE /OX".

TT-DOS

TT-DOS is an older Czech program, that contains a DOS 2.5 clone FMS, and a very first version of the "B:" handler. The format of tape-files is compatible with B-TAPE. Since TT-DOS is a DOS 2.5 alike system, no filenames - including the ones on tape - may contain the character "_", and also numbers are not allowed at the first position. TT-DOS doesn't support the random numbers, so every tape-files under TT-DOS should have different names. The functions of "B:" handler found in TT-DOS are also supported by B-TAPE. B-TAPE does support some new functions (NOTE, POINT, Binary load, and the access at SIO-level), that are not available under TT-DOS. A small difference is also in the format of directory-listing. TT-DOS uses the character "#" for files with double-blocks (B-TAPE uses "*"), and the "FREE SECTORS" counter is DOS 2.5 alike, so it shows "999+FREE SECTORS".

FOR SALE

Black Box with Floppy Board enhancement, Hard Drive 40MB formatted and partitioned. 3 PBI Drives: 1: 3½ 1.4MB 1:5¼ 360K, 1:5¼ 1.2MB. Citizen 120D+ with 14 spare re-inked Ribbons. Modem WS4000 Miracle Technology. All cables to connect to Black Box to run parallel, plus all manuals and power units. If wanted will also sell a 1Meg 65XE, setup for Black Box. Sell as one item, price £400, without computer £350. Buyer must collect or pay Postage. I include utilities and a spare 3½ 720K drive.

All these items are surplus to requirement, I still have an 800XL with 256K memory and disk drives.

Black Box when purchased cost £319 + £72 VAT, custom charges and import duty.

Contact Max: on

0191 - 586 6795

REVIEWS

BY KEVIN COOKE

OK, here we go with some more reviews. And I'm late sending them to the guys at TWAUG again! Ooops! Luckily, I've now finished my first-year exams so all I have to do is worry continuously until the results arrive!!! Oh well, enough chit-chat, let's take a look at some more games.

Title: INSIDE

Sold by: Micro Discount,

265 Chester Road,

Streetly,

West Midlands B74 3EA,

ENGLAND.

Tel: 0121- 353 5730

Price: 5 Pounds 95p (+ P&P)

It's rare that a unique piece of serious computer software it released. However, it's even rarer that a truely unique computer game is released. Inside changes that!

The story goes that your Atari has been over-run by a computer virus of catastrophic proportions - it can literally replace components of various IC's with incorrect parts, causing the eventual death of your beloved machine. As an electrician, your task is to go inside of the computer (literally!) and repair the broken IC's.

Luckily, you are supplied with the latest in computer infiltration craft - a high-tech spacecraft capable of manouvering at high speeds.

The first thing to appear when the game is loaded is the title screen. Here a digitised voice says "Tech Tech Technology, Technology" before a fairly pleasant tune begins. The title screen almost looks like a demo with the word "inside" looping around a large computer chip. Pressing FIRE, as usual, starts the main game.

The game screen is divided into two playfields, each of which contains a spaceship, allowing you to play a simultaneous two player game. At the bottom of each screen are numerous parts of the computer - touching these will make you lose energy. However, amongst these parts are bits which may need to be repaired - you'll know which bits need repairing from the messages which will appear on screen but it's your task to find out where these parts actually are! The screen scrolls so that there are actually about 4 screen-lengths of computer

REVIEWS

parts.

When you do find a component which needs repairing and you land on it, the screen changes to one showing a circuit diagram-type picture with the faulty components flashing. You must then press fire and replace these bits with a choice of around 25 other parts. It's a difficult concept to explain but it's actually quite straightforward - honest!

To make matters more complicated (but realistic), when certain IC's are not repaired, the program is affected in the correct way. For example, if the POKEY chip is broken, strange sounds will be heard..... if the PIA is broken, your control of your spaceship will become more difficult, etc.

INSIDE is certainly unusual. This could be it's failing for people who are not prepared to experiment with the game and would prefere a simple shoot 'em up. However, perserverance will show that INSIDE is unique, graphically and musically pleasing and above all, fun to play. I recommend it!

Title: BATTLE SHIPS

Sold at: Micro Discount (see above for address)

Price: 4 Pounds (+ P&P)

This game is a clone of the game which has been around on the ST an AMIGA for several years now. But do you really want to play battleships on your computer?

The game, after taking quite a while to load presents you with three options - a one player game using one joystick or a two player game using either one of two joysticks. After selecting which type of game to play, the real fun begins.

First of all, you must place your ships on the large grid - you can move them around and rotate them with the aid of your joystick. In a two player game, each player would obviously have to leave the room whilst the other places his ship. When both players have done this, each can take it in turn to aim their shots at several of the grid square. Each player starts off being able to fire 20 shots but this number decreases as more of his ships are sunk.

After placing your shots, a different scren appears showing the guns on a ship shooting at the water in the

REVIEWS

distance. A hit is denoted by a distant explosion whilst a miss is denoted by a simple splash of water. This process is repeated several times until one player's ships have all been destroyed. At thIS point, a message appears telling the players who's navy won the battle.

There is nothing particularly exciting about the game. You'd probably have as much fun playing it with a normal magnetic set - the sort available in most toy shops. However, if you have no friends (saddo!) or don't want to fuss around with getting those magnetic pieces to stick, the game could be just for you. It's actually well done but, to be honest, if BATTLESHIPS has never turned you on before than this is unlikely to do so either.

Overall, if you like BATTLESHIPS then this is ideally priced. If not, stay clear.

Title: FATUM

Sold at: Micro Discount (see above for address)

Price: 4 Pounds (+ P&P)

FATUM is a new shoot 'em up from ASF of Poland.

The disk, when first loaded, displays a menu from which you have the option of displaying a introduction demo or starting the game.

The demo looks rather nice. A paralex scrolling landscape appears at the bottom of the screen in numerous colours and eventually a (bulky) spaceship flies into view. Text (in polish) is then displayed on the screen. Although this demo looks nice, it does seem a little pointless. So what's the game like?

Well, the title screen also looks a little like a demo. Pressing fire brings up a message saying "ETAP 1" (level 1) and a music tune plays... and plays... and plays... and you can't skip it! This got the game off to a bad start in my books.

The actual game is a vertically scrolling shoot 'em up. Graphically there are no major problems (except for the design of your ship which I don't think looks particularly exciting!). However, gameplay's where it count so what is FATUM like, I hear you ask. To be perfectly honest, not too good!

FATUM is a poor imitation of SIDEWINDER (or should that be SIDEWINDER II?). Control of the ship is very responsive but most of

REVIEWS

the time, perhaps too responsive. I realise that movement needed to be fast to steer the ship through the tiny gaps but why were the gaps made so tiny in the first place?!! Aliens do attack in some nice formations but even your laser cannon fails to raise the blood pressure. Something MUST be wrong if gratuitous destruction can do nothing to excite me!

It is with regret that I do not feel that I can recommend this game at all. It may please die-hard fans of shoot 'em up's but for most other people, it will probably not excite in the slightest.

I'm left wishing that the main game was vertically scrolling and graphically more like the demo. Alas, it's not and as such would probably be a buy that you would regret. Save your money and buy someting else instead.

Sorry the column's a bit shorter this time. Expect it to be back in it's former glory next issue with even more reviews. In the meantime, support your Atari dealers - they need you as much as you need them!

MAIL BAG

May 28, 1996

Dear T.W.A.U.G.

I am mailing you a game called "SUB ATTACK". It is an older game but I reworked it so that it now saves your name and high score in that game and on the disk. It is a very playable game. We use it when we hold a contest in the club. Alex modified the high score routine in the game. I think you will like it.

On the back of the disk we have a very nice "COOKING RECIPE PROGRAM" which Alex and I also modified to work better.

Our best regards to you.

Atarily,

Ron Fetzer, Secretary & Treasurer,
OL' HACKERS ATARI USER GROUP INC.

Reply by Max:

We do appreciate your kindness in sending us the disk with the programs, mentioned above, in your letter.

John thought it a good idea to share the game and cooking program with our readers and therefore he put these two programs onto the B side of this issue disk.

MAIL BAG

June 2, 1996

Dear Max and John,

I received your latest newsletter issue #21, but I'm sorry to inform you that the disk must have been damaged in transit and would not load on either side. Would you please be good enough to send me a replacement for our PD library.

A suggestion if you don't mind. I think that your last newsletter magazine has too many articles that are very technical in nature, except of course the article on BASIC and the games review. I think you should consider a broader spectrum to cover those 8 BIT folks who are sort of above the basics, but below the techies, but interested in utilities, DTP, etc. This is just my thoughts, which I wanted to share with you.

I still think you fellows put out a great newsletter duo.

Atarily,

Alex Pignato
President
OL' HACKERS ATARI USER GROUP INC.

Reply by Max:

I am sorry to hear about the damaged issue disk you've received, it does happen from time to time, I have received damaged disks, mailed locally.

Thank you Alex for your suggestion, or shall I call it a criticism, I am really pleased about it, because I thought everybody was satisfied with the content of the newsletter. You see we haven't received any new articles for quite a while, the upgrade articles were the only ones we had received, but I am pleased to say this issue contains the last of this kind of material. I also found these articles to technical.

It is rather a bit difficult when no outside help is forthcoming, especially with me having had to go back into hospital for another operation. Since I had this op, which was on the 6th of June, I haven't been able to concentrate properly. After the op., something's gone wrong and left me with a painful chest.

I do hope that we will receive some new material for the next issue, or suggestions what the readers would like to see. I am thinking to include an article on DLI.

MAIL BAG

5th June 1996

Dear TWAUG,

Congratulations on another super Newsletter, the new format is much easier to read. Perhaps, on looking back at the last few issues, I prefer the general layout of issue 18 which did'nt have the border but had the line separating the two columns. How about alternating the two styles.

I was disappointed at missing SAMS but the first I knew about the change of venue and the date was in Page 6 which arrived on May 11th, just a month after the event. There must be many others in the same position and this, no doubt, accounts for the poor attendance. I will endeavour to be at the next show which I assume will be in Stafford in the rain in November.

Enclosed is an order for PD disks for your kind attention..

Very best wishes,

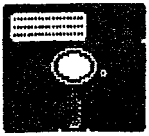
Leslie Benson

Reply by Max

Thank you for your letter and I am pleased that the new format is easier to read. As I mentioned in my previous reply, I like to hear suggestions and ideas on how to improve the newsletter.

I have already started including centre lines, where a new article begins. The reason I have put a border line on each page is for a guide when I cut the pages to size. All issues up to 18 were printed in the A4 size and I didn't have to cut the pages to the A5 size. This new format is printed in the A5 size, as you see it onto A4 size paper and it needs cutting after it is printed. Without the border lines I find it a bit difficult to cut all pages to the right size, with the border lines on I find it much easier and quicker to cut. I will of course, as you suggest, alternate the issue with different borders and centre lines.

TWAUG was at the spring show in Spalding, Lincs., I myself wasn't there only John. The venue for SAMS will be in Stafford again on the 9th November, and I have no doubt, it will be raining as well.



DISK CONTENT

Side A of this issue disk contains two basic games and three screen dump programs, a font maker program and a character set data maker, an Atari file coder and a password autoboot maker. On Side B we've put the programs we've received from our friends from New York. A game called Subattack and a Recipe program.

The Screen Dumps are for the Epson compatible printers, a NP-10 and the Okimate. These programs let you print pictures in mono, colour, in Graphics 8 or Graphics 9. When the program is loaded you have the Option the read the instructions by just pressing the space bar.

The Font Maker program lets you design your own fonts which can be save to disk, it also lets you view a font set which you can load in.

The Character Set Datamaker creates a set of Data Statements from a saved character set. The Data Statement will be written to disk as a LIST File, the Data can then be merged with your program using the ENTER command.

Have you ever wanted to protect your special files from prying eyes? Then Atari File Coder is the program, it uses a form of password protection. Without the correct password no one can recover the files (not even you). The same program is used to decode the files, as long as you enter the correct password. The instructions are on the disk, under the filename -Atari File Coder, it can be read from the menu.

When you run the Password Autoboot Maker it writes an AUTORUN.SYS onto the disk.

The two games are written in Basic. NYPY, I found it rather difficult to master, you must guide a funny looking guy, through an under ground maze.

SAGUARO, the aim in this game is to pick up as many eggs as you can which a big bird is laying all over the place. But beware, if this bird catches up with you, then you've had it.

TWAUG NEWSLETTER

ADVERTISING USER GROUPS

LACE

The London Atari Computer Enthu-siasts

As a member of LACE you will receive a monthly newsletter and have access to a monthly meeting. They also support the ST and keep a large selection of ST and 8-bit PD software.

The membership fee is
£8.00 annually
for more information contact:
Mr. Roger Lacey
LACE Secretary
41 Henryson Road
Crofton Park
London SE4 1HL
Tel.: 0181 - 690 2548

O.H.A.U.G.

The OL'HACKERS ATARI USER GROUP INC.

O.H.A.U.G. is an all 8-bit user group in the STATE of NEW YORK.

They are producing a bi-monthly double sided disk based newsletter. The disk comes with its own printing utility, which lets you read the content of the disk, one screen page at the time, and/or you can make a hard copy of the disk, in one, two or three columns and 6 to 8 lines to the inch. A large PD Library is available.

Contact:
Mr.A.Pignato
O.H.A.U.G.
3376 Ocean Harbor Drive
Oceanside, N.Y. 11572
USA

ADVERTISEMENT

CHAOSI COMPUTERS

PO BOX 30

MANCHESTER M19 2DX

Telephone: (0161) 737 1946

THE HYPER DRIVE

Upgrade your *ATARI 1050* disk drive with a **HYPER DRIVE** enhancement from **CHAOSI COMPUTERS.**

The **HYPER DRIVE** is an easily installed hardware & software package for the *ATARI 1050* which will enable your disk drive to back-up most disks protected by unreadable or badly formatted sectors. Most copied disks can then be loaded on any 1050, whether enhanced with a **HYPER DRIVE** or not. The **HYPER DRIVE** enhancement also offers fast reading, writing, formatting and copying in single, medium or true double density formats (i,e. it is compatible with *RANA, PERCOM* and *INDUS* double density drives, and will read *U.S. DOUBLER* type format).

Fitting the **HYPER DRIVE** coudn't be simpler and requires no special tools or soldering. It simply plugs into socket on the 1050 circuit board. And with our VERSION II software package and full 28 page manual, it is one of the most versatile disk drive enhancements /copiers you can buy. **HYPER DRIVEs** are available exclusively from **CHAOSI COMPUTERS** at a special intro-ductory price of just £30.00 each.

Please make Cheques/Postal Orders payable to **'P. HOLLINS'.**

Prices are subject to change, from time to time, due to component costs, so wherever possible please *'phone to check'.*

TWAUG NEWSLETTER

MICRO-DISCOUNT

Offers
The complete Mail Order
service for Atari 8 Bit
XL/XE users

ATARI ATARI

4 th September 1995

